Package 'lrd'

May 8, 2026

Title A Package for Processing Lexical Response Data

Version 0.1.0

Description Lexical response data is a package that
can be used for processing cued-recall, free-recall,
and sentence responses from memory experiments.

Depends R (>= 3.5.0)

Imports stats, utils, knitr

Suggests ggplot2, rmarkdown, reshape

License LGPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

VignetteBuilder knitr

URL https://npm27.github.io/lrd/

NeedsCompilation no

Author Nicholas Maxwell [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4517-8323>),
Erin M. Buchanan [aut] (ORCID: <https://orcid.org/0000-0002-9689-4189>)

Maintainer Nicholas Maxwell <nicholas.maxwell@usm.edu>

Repository CRAN

Date/Publication 2021-12-09 09:50:02 UTC

Contents

`answer_key_free` *Answer Key Example Data*

Description

Dataset that includes the answer key for free recall data. Pair with the wide_data dataset for examples.

Usage

```
data(answer_key_free)
```

Format

A data frame of answers for a free recall test

Answer_Key: a list of free recall answers

answer_key_free2 *Answer Key Example Data*

Description

Dataset that includes the answer key for free recall data. Pair with the free_data dataset for examples.

Usage

```
data(answer_key_free2)
```

Format

A data frame of answers for a free recall test

Answer_Key: a list of free recall answers

arrange_data *Arrange Data for Free Recall Scoring*

Description

This function takes wide format free recall data where all responses are stored in the same cell and converts it to long format.

Usage

```
arrange_data(data, responses, sep, id, repeated = NULL)
```

Arguments

data	a dataframe of the variables you would like to return. Other variables will be included in the returned output in long format if they represent a one to one match with the participant ID. If you have repeated data, please use the repeated argument or run this function several times for each trial.
responses	a column name in the dataframe that contains the participant answers for each item in quotes (i.e., "column")
sep	a character separating each response in quotes - example: ",".
id	a column name containing participant ID numbers from the original dataframe
repeated	(optional) a single column name or set of columns that indicate repeated measures columns you would like to keep with the data. You should include all columns that are not a one to one match with the subject ID (i.e., participants saw multiple trials). Please see our vignette for an example.

Value

A dataframe of the participant answers including:

Sub.ID	The participant id number
response	The participant response
position	The position number of the response listed
other	Any additional columns included

Examples

```
#This dataset includes a subject number, set of answers, and
#experiment condition.

data(wide_data)

DF_long <- arrange_data(
 data = wide_data,
 responses = "Response",
 sep = ",",
 id = "Sub.ID")

head(DF_long)
```

---

crp *Conditional Response Probability*

Description

This function calculates the conditional response probability of each lag position. Participants' lag between subsequent named items is tallied and then divided by the possible combination of subsequent lags given their response pattern.

Usage

```
crp(data, position, answer, id, key, scored)
```

Arguments

data	a dataframe of the scored free recall that you would like to calculate - use prop_correct_free() for best formatting.
position	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
answer	a column name of the answer given for that position in the original dataframe.
id	a column name of the participant id in the original dataframe.

key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.
scored	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).

Details

This output can then be used to create a CRP visualizations, and an example can be found in our manuscript/vignettes.

Important: The code is written assuming the data provided are for a single recall list. If repeated measures are used (i.e., there are multiple lists completed by each participant or multiple list versions), you should use this function several times, once on each list/answer key.

Value

DF_CRP	A dataframe of the proportion correct for each conditional lag position including any other between subjects variables present in the data.

Examples

```
data(free_data)
data(answer_key_free2)

free_data <- subset(free_data,
 List_Type == "Cat_Recall_L1")

DF_long <- arrange_data(data = free_data,
 responses = "Response",
 sep = " ",
 id = "Username")

scored_output <- prop_correct_free( data = DF_long,
 responses = "response",
 key = answer_key_free2$Answer_Key,
 id = "Sub.ID",
 cutoff = 1,
 flag = TRUE,
 group.by = "Version")

crp_output <- crp(data = scored_output$DF_Scored,
 position = "position",
 answer = "Answer",
 id = "Sub.ID",
 key = answer_key_free2$Answer_Key,
 scored = "Scored")

 head(crp_output)
```

`crp_multiple` *Conditional Response Probability for Multiple Lists*

Description

This function calculates the conditional response probability of each lag position. Participants' lag between subsequent named items is tallied and then divided by the possible combination of subsequent lags given their response pattern. This function was designed to handle multiple or randomized lists across participants.

Usage

```
crp_multiple(data, position, answer, id, key, key.trial, id.trial, scored)
```

Arguments

`data`	a dataframe of the scored free recall that you would like to calculate - use prop_correct_free() for best formatting.
`position`	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
`answer`	a column name of the answer given for that position in the original dataframe.
`id`	a column name of the participant id in the original dataframe.
`key`	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.
`key.trial`	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
`id.trial`	a column name containing the trial numbers for the participant data from the original dataframe. Note that the free response "key" trial and this trial number should match. The trial key will be repeated for each answer a participant gave.
`scored`	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).

Details

This output can then be used to create a CRP visualizations, and an example can be found in our manuscript/vignettes.

Value

`DF_CRP`	A dataframe of the proportion correct for each conditional lag position including any other between subjects variables present in the data.

Examples

```
data("multi_data")
data("multi_answers")

DF_long <- arrange_data(data = multi_data,
                        responses = "Response",
                        sep = " ",
                        id = "Sub.ID",
                        repeated = "List.Number")

library(reshape)
multi_answers$position <- 1:nrow(multi_answers)
answer_long <- melt(multi_answers,
                    measured = colnames(multi_answers),
                    id = "position")
colnames(answer_long) <- c("position", "List.ID", "Answer")

answer_long$List.ID <- gsub(pattern = "List",
                            replacement = "",
                            x = answer_long$List.ID)

DF_long$response <- tolower(DF_long$response)
answer_long$Answer <- tolower(answer_long$Answer)
answer_long$Answer <- gsub(" ", "", answer_long$Answer)

scored_output <- prop_correct_multiple(data = DF_long,
                                       responses = "response",
                                       key = answer_long$Answer,
                                       key.trial = answer_long$List.ID,
                                       id = "Sub.ID",
                                       id.trial = "List.Number",
                                       cutoff = 1,
                                       flag = TRUE)

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)


crp_output <- crp_multiple(data = scored_output$DF_Scored,
                           key = answer_long$Answer,
                           position = "position",
                           scored = "Scored",
                           answer = "Answer",
                           id = "Sub.ID",
                           key.trial = answer_long$List.ID,
                           id.trial = "List.Number")

 head(crp_output)
```

cued_data *Cued Recall Data*

Description

Dataset that includes cued recall data in long format. Participants were given a cue, and they were required to remember the response listed in the dataset. This dataset is in long format, which is required for most functions.

Usage

```
data(cued_data)
```

Format

A data frame of answers for a cued recall test data

id: the participant id trial: the trial id response: the response the participant gave to the cue key: the answer for this trial id condition: the between subjects group the participants were in

cued_data_groupby *Cued Recall Data with Multiple Conditions*

Description

Dataset that includes cued recall data in long format. Participants were given a cue, and they were required to remember the response listed in the dataset. This dataset is in long format, which is required for most functions.

Usage

```
data(cued_data_groupby)
```

Format

A data frame of answers for a cued recall test data

Subject: the participant id Target: the answer for this trial id Response: the response the participant gave to the cue Condition: the between subjects group the participants were in Condition2: the second between subjects group the participants were in

cued_recall_manuscript *Cued Recall Data from Manuscript*

Description

Dataset that includes cued recall data in long format. Participants were given a cue, and they were required to remember the response listed in the dataset. This dataset is in long format, which is required for most functions.

Usage

```
data(cued_data)
```

Format

A data frame of answers for a cued recall test data

Sub.ID: the participant id Trial_num: the trial id Cue: the cue shown to participants Target: the answer for this trial id Answer: the participant answer for this trial

free_data *Free Recall Data*

Description

Dataset that includes free recall data in long format. Participants were given a list of words to remember, and then asked to recall the words. This dataset is in wide format, which should be converted with arrange data.

Usage

```
data(free_data)
```

Format

A data frame of answers for a free recall test data

Username: the participant id List_Types: a repeated measures condition participants were in Response: the response the participant gave to the cue Version: the version of the list_type given Batch: the batch of participants that were run together

kappa *Cohen's Kappa*

Description

This function returns Cohen's Kappa k for two raters. Kappa indicates the inter-rater reliability for categorical items. High scores (closer to one) indicate agreement between raters, while low scores (closer to zero) indicate low agreement between raters. Negative numbers indicate they don't agree at all!

Usage

```
kappa(rater1, rater2, confidence = 0.95)
```

Arguments

rater1	Rater 1 scores or categorical listings
rater2	Rater 2 scores or categorical listings
confidence	Confidence interval proportion for the kappa interval estimate. You must supply a value between 0 and 1.

Details

Note: All missing values will be ignored. This function calculates kappa for 0 and 1 scoring. If you pass categorical variables, the function will return a percent match score between these values.

Value

p_agree	Percent agreement between raters
kappa	Cohen's kappa for yes/no matching
se_kappa	Standard error for kappa wherein standard error is the square root of: (agree * (1-agree)) / (N * (1 - random agreement)^2)
kappa_LL	Lower limit for the confidence interval of kappa
kappa_UL	Upper limit for the confidence interval of kappa

Examples

```
#This dataset includes two raters who wrote the word listed by
#the participant and rated if the word was correct in the recall
#experiment.

data(rater_data)

#Consider normalizing the text if raters used different styles
#Calculate percent match for categorical answers
kappa(rater_data$rater1_word, rater_data$rater2_word)
```

```
kappa(rater_data$rater1_score, rater_data$rater2_score)
```

`multi_answers` *Answer Key Example Data for Multiple Lists*

Description

Dataset that includes the answer key for free recall data. Pair with the multi_data dataset for examples.

Usage

```
data(multi_answers)
```

Format

A data frame of answers for a free recall test

List1: a list of free recall answers List2: a second list of free recall answers etc.

`multi_data` *Free Recall Data in Wide Format with Multiple Lists*

Description

Dataset that includes free recall data in long format. Participants were given a list of words to remember, and then asked to recall the words. This dataset is in wide format, which should be converted with arrange data.

Usage

```
data(multi_data)
```

Format

A data frame of answers for a free recall test data

Sub.ID: the participant id List.Type: the type of list a person saw Response: the response the participant gave to the cue List.Number: the number of the list they completed

pfr *Probability of First Recall*

Description

This function calculates the probability of first recall for each serial position. The total number of times an item was recalled first is divided by the total number of first recalls (i.e., the number of participants who wrote anything down!).

Usage

```
pfr(data, position, answer, id, key, scored, group.by = NULL)
```

Arguments

data	a dataframe of the scored free recall that you would like to calculate - use prop_correct_free() for best formatting.
position	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
answer	a column name of the answer given for that position in the original dataframe.
id	a column name of the participant id in the original dataframe.
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.
scored	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

This output can then be used to create a PFR visualizations, and an example can be found in our manuscript/vignettes.

Important: The code is written assuming the data provided are for a single recall list. If repeated measures are used (i.e., there are multiple lists completed by each participant or multiple list versions), you should use this function several times, once on each list/answer key.

Value

DF_PFR	A dataframe of the probability of first response for each position including group by variables if indicated.

Examples

```
data(free_data)
data(answer_key_free2)

free_data <- subset(free_data,
 List_Type == "Cat_Recall_L1")

DF_long <- arrange_data(data = free_data,
 responses = "Response",
 sep = " ",
 id = "Username")

scored_output <- prop_correct_free(data = DF_long,
 responses = "response",
 key = answer_key_free2$Answer_Key,
 id = "Sub.ID",
 cutoff = 1,
 flag = TRUE,
 group.by = "Version")

pfr_output <- pfr(data = scored_output$DF_Scored,
 position = "position",
 answer = "Answer",
 id = "Sub.ID",
 key = answer_key_free2$Answer_Key,
 scored = "Scored",
 group.by = "Version")

 head(pfr_output)
```

---

pfr_multiple *Probability of First Recall for Multiple Lists*

Description

This function calculates the probability of first recall for each serial position. The total number of times an item was recalled first is divided by the total number of first recalls (i.e., the number of participants who wrote anything down!).

Usage

```
pfr_multiple(
  data,
  position,
  answer,
  id,
  key,
  key.trial,
```

```
  id.trial,
  scored,
  group.by = NULL
)
```

Arguments

data	a dataframe of the scored free recall that you would like to calculate - use prop_correct_free() for best formatting.
position	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
answer	a column name of the answer given for that position in the original dataframe.
id	a column name of the participant id in the original dataframe.
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.
key.trial	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
id.trial	a column name containing the trial numbers for the participant data from the original dataframe. Note that the free response "key" trial and this trial number should match. The trial key will be repeated for each answer a participant gave.
scored	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

This output can then be used to create a PFR visualizations, and an example can be found in our manuscript/vignettes.

Value

DF_PFR	A dataframe of the probability of first response for each position including group by variables if indicated.

Examples

```
data("multi_data")
data("multi_answers")

DF_long <- arrange_data(data = multi_data,
                        responses = "Response",
```

```
                        sep = " ",
                        id = "Sub.ID",
                        repeated = "List.Number")

library(reshape)
multi_answers$position <- 1:nrow(multi_answers)
answer_long <- melt(multi_answers,
                    measured = colnames(multi_answers),
                    id = "position")
colnames(answer_long) <- c("position", "List.ID", "Answer")

answer_long$List.ID <- gsub(pattern = "List",
                            replacement = "",
                            x = answer_long$List.ID)

DF_long$response <- tolower(DF_long$response)
answer_long$Answer <- tolower(answer_long$Answer)
answer_long$Answer <- gsub(" ", "", answer_long$Answer)

scored_output <- prop_correct_multiple(data = DF_long,
                                    responses = "response",
                                    key = answer_long$Answer,
                                    key.trial = answer_long$List.ID,
                                    id = "Sub.ID",
                                    id.trial = "List.Number",
                                    cutoff = 1,
                                    flag = TRUE)

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)

head(scored_output$DF_Group)

pfr_output <- pfr_multiple(data = scored_output$DF_Scored,
                           key = answer_long$Answer,
                           position = "position",
                           scored = "Scored",
                           answer = "Answer",
                           id = "Sub.ID",
                           key.trial = answer_long$List.ID,
                           id.trial = "List.Number")

 head(pfr_output)
```

`prop_correct_cued` *Proportion Correct Cued Recall*

Description

This function computes the proportion of correct responses per participant. Proportions can either be separated by condition or collapsed across conditions. You will need to ensure each trial is marked with a unique id to correspond to the answer key.

Usage

```
prop_correct_cued(
  data,
  responses,
  key,
  key.trial,
  id,
  id.trial,
  cutoff = 0,
  flag = FALSE,
  group.by = NULL
)
```

Arguments

data	a dataframe of the variables you would like to return. Other variables will be included in the scored output and in the participant output if they are a one to one match with the participant id.
responses	a column name in the dataframe that contains the participant answers for each item in quotes (i.e., "column")
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe.
key.trial	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
id	a column name containing participant ID numbers from the original dataframe.
id.trial	a column name containing the trial numbers for the participant data from the original dataframe.
cutoff	a numeric value that determines the criteria for scoring (i.e., 0 = strictest, 5 = is most lenient). The scoring criteria uses a Levenshtein distance measure to match participant responses to the answer key.
flag	a logical argument if you want to flag participant scores that are outliers using z-scores away from the mean score for group
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

Note: other columns included in the dataframe will be found in the final scored dataset. If these other columns are between subjects data, they will also be included in the participant dataset (i.e., there's a one to one match of participant ID and column information).

Value

DF_Scored	The dataframe of the original response, answer, scoring, and any other or grouping variables. This dataframe can be used to determine if the cutoff score and scoring matched your answer key as intended. Distance measures are not perfect! Issues and suggestions for improvement are welcome.
DF_Participant	A dataframe of the proportion correct by participant, which also includes optional z-scoring, grouping, and other variables.
DF_Group	A dataframe of the summary scores by any optional grouping variables, along with overall total proportion correct scoring.

Examples

```
#This data contains cued recall test with responses and answers together.
#You can use a separate answer key, but this example will show you an
#embedded answer key. This example also shows how you can use different
#stimuli across participants (i.e., each person sees a randomly selected
#set of trials from a larger set).

data(cued_data)

scored_output <- prop_correct_cued(data = cued_data,
 responses = "response",
 key = "key",
 key.trial = "trial",
 id = "id",
 id.trial = "trial",
 cutoff = 1,
 flag = TRUE,
 group.by = "condition")

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)

head(scored_output$DF_Group)
```

---

prop_correct_free *Proportion Correct Free Recall*

Description

This function computes the proportion of correct responses per participant. Proportions can either be separated by condition or collapsed across conditions.

Usage

```
prop_correct_free(
  data,
  responses,
  key,
  id,
  cutoff = 0,
  flag = FALSE,
  group.by = NULL
)
```

Arguments

data	a dataframe of the variables you would like to return. Other variables will be included in the scored output and in the participant output if they are a one to one match with the participant id.
responses	a column name in the dataframe that contains the participant answers for each item in quotes (i.e., "column")
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe.
id	a column name containing participant ID numbers from the original dataframe
cutoff	a numeric value that determines the criteria for scoring (i.e., 0 = strictest, 5 = is most lenient). The scoring criteria uses a Levenshtein distance measure to match participant responses to the answer key.
flag	a logical argument if you want to flag participant scores that are outliers using z-scores away from the mean score for group
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

Note: other columns included in the dataframe will be found in the final scored dataset. If these other columns are between subjects data, they will also be included in the participant dataset (i.e., there's a one to one match of participant ID and column information).

Value

DF_Scored	The dataframe of the original response, answer, scoring, and any other or grouping variables. This dataframe can be used to determine if the cutoff score and scoring matched your answer key as intended. Distance measures are not perfect! Issues and suggestions for improvement are welcome.

DF_Participant A dataframe of the proportion correct by participant, which also includes optional z-scoring, grouping, and other variables.

DF_Group A dataframe of the summary scores by any optional grouping variables, along with overall total proportion correct scoring.

Examples

```
data(wide_data)
data(answer_key_free)

DF_long <- arrange_data(data = wide_data,
 responses = "Response",
 sep = ",",
 id = "Sub.ID")

scored_output <- prop_correct_free(data = DF_long,
 responses = "response",
 key = answer_key_free$Answer_Key,
 id = "Sub.ID",
 cutoff = 1,
 flag = TRUE,
 group.by = "Disease.Condition")

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)

head(scored_output$DF_Group)
```

---

prop_correct_multiple *Proportion Correct Free Recall for Multiple Lists*

Description

This function computes the proportion of correct responses per participant. Proportions can either be separated by condition or collapsed across conditions. This function extends prop_correct_free() to include multiple or randomized lists for participants.

Usage

```
prop_correct_multiple(
  data,
  responses,
  key,
  key.trial,
  id,
  id.trial,
  cutoff = 0,
```

```
  flag = FALSE,
  group.by = NULL
)
```

Arguments

data	a dataframe of the variables you would like to return. Other variables will be included in the scored output and in the participant output if they are a one to one match with the participant id.
responses	a column name in the dataframe that contains the participant answers for each item in quotes (i.e., "column")
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe.
key.trial	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
id	a column name containing participant ID numbers from the original dataframe.
id.trial	a column name containing the trial numbers for the participant data from the original dataframe. Note that the free response "key" trial and this trial number should match. The trial key will be repeated for each answer a participant gave.
cutoff	a numeric value that determines the criteria for scoring (i.e., 0 = strictest, 5 = is most lenient). The scoring criteria uses a Levenshtein distance measure to match participant responses to the answer key.
flag	a logical argument if you want to flag participant scores that are outliers using z-scores away from the mean score for group
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

Note: other columns included in the dataframe will be found in the final scored dataset. If these other columns are between subjects data, they will also be included in the participant dataset (i.e., there's a one to one match of participant ID and column information).

Value

DF_Scored	The dataframe of the original response, answer, scoring, and any other or grouping variables. This dataframe can be used to determine if the cutoff score and scoring matched your answer key as intended. Distance measures are not perfect! Issues and suggestions for improvement are welcome.
DF_Participant	A dataframe of the proportion correct by participant, which also includes optional z-scoring, grouping, and other variables.
DF_Group	A dataframe of the summary scores by any optional grouping variables, along with overall total proportion correct scoring.

Examples

```
data("multi_data")
data("multi_answers")

DF_long <- arrange_data(data = multi_data,
                        responses = "Response",
                        sep = " ",
                        id = "Sub.ID",
                        repeated = "List.Number")

library(reshape)
multi_answers$position <- 1:nrow(multi_answers)
answer_long <- melt(multi_answers,
                    measured = colnames(multi_answers),
                    id = "position")
colnames(answer_long) <- c("position", "List.ID", "Answer")

answer_long$List.ID <- gsub(pattern = "List",
                            replacement = "",
                            x = answer_long$List.ID)

DF_long$response <- tolower(DF_long$response)
answer_long$Answer <- tolower(answer_long$Answer)
answer_long$Answer <- gsub(" ", "", answer_long$Answer)

scored_output <- prop_correct_multiple(data = DF_long,
                                       responses = "response",
                                       key = answer_long$Answer,
                                       key.trial = answer_long$List.ID,
                                       id = "Sub.ID",
                                       id.trial = "List.Number",
                                       cutoff = 1,
                                       flag = TRUE)

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)
```

---

prop_correct_sentence *Proportion Correct for Sentences*

Description

This function computes the proportion of correct sentence responses per participant. Proportions can either be separated by condition or collapsed across conditions. You will need to ensure each trial is marked with a unique id to correspond to the answer key.

Usage

```
prop_correct_sentence(
  data,
  responses,
  key,
  key.trial,
  id,
  id.trial,
  cutoff = 0,
  flag = FALSE,
  group.by = NULL,
  token.split = " "
)
```

Arguments

data	a dataframe of the variables you would like to return. Other variables will be included in the scored output and in the participant output if they are a one to one match with the participant id.
responses	a column name in the dataframe that contains the participant answers for each item in quotes (i.e., "column")
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe.
key.trial	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
id	a column name containing participant ID numbers from the original dataframe
id.trial	a column name containing the trial numbers for the participant data from the original dataframe
cutoff	a numeric value that determines the criteria for scoring (i.e., 0 = strictest, 5 = is most lenient). The scoring criteria uses a Levenshtein distance measure to match participant responses to the answer key.
flag	a logical argument if you want to flag participant scores that are outliers using z-scores away from the mean score for group
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns
token.split	an optional argument that can be used to delineate how to separate tokens. The default is a space after punctuation and additional spacing is removed.

Details

Note: other columns included in the dataframe will be found in the final scored dataset. If these other columns are between subjects data, they will also be included in the participant dataset (i.e., there's a one to one match of participant ID and column information).

Value

DF_Scored	The dataframe of the original response, answer, scoring, and any other or grouping variables. This dataframe can be used to determine if the cutoff score and scoring matched your answer key as intended. Distance measures are not perfect! Issues and suggestions for improvement are welcome.
DF_Participant	A dataframe of the proportion correct by participant, which also includes optional z-scoring, grouping, and other variables.
DF_Group	A dataframe of the summary scores by any optional grouping variables, along with overall total proportion correct scoring.

Examples

```
#This data contains sentence recall test with responses and answers together.
#You can use a separate answer key, but this example will show you an
#embedded answer key. This example also shows how you can use different
#stimuli across participants (i.e., each person sees a randomly selected
#set of trials from a larger set).

data(sentence_data)

scored_output <- prop_correct_sentence(data = sentence_data,
 responses = "Response",
 key = "Sentence",
 key.trial = "Trial.ID",
 id = "Sub.ID",
 id.trial = "Trial.ID",
 cutoff = 1,
 flag = TRUE,
 group.by = "Condition",
 token.split = " ")

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)

head(scored_output$DF_Group)
```

---

rater_data *Rater Data*

Description

Dataset that contains scoring and ratings for a recall test that was rated by two raters. Use with the kappa function as an example.

Usage

```
data(rater_data)
```

Format

A data frame of scored answers for inter-rater reliability

Sub.ID: the participant id rater1_word: the word choice for the subject the rater selected rater1_score: the score for the participant given by the rater rater2_word: the word choice for the subject the rater selected rater2_score: the score for the participant given by the rater

---

`sentence_data` *Sentence Recall Data*

Description

Dataset that includes sentence recall data in long format. Participants were given a sentence to remember, and then asked to recall the words. This dataset is in long format, which is required for these functions.

Usage

```
data(sentence_data)
```

Format

A data frame of answers for a sentence recall test data

Sub.ID: the participant id Trial.ID: the id for the trial given to participant Sentence: the answer to the trial that the participant should have given Response: the response the participant gave to that trial Condition: the between subjects condition the participant was in

---

`serial_position` *Serial Position Calculator*

Description

This function calculates the proportion correct of each item in the serial position curve. Data should include the participant's answers in long format (use arrange_data() in this package for help), the answer key of the items in order, and a column that denotes the order a participant listed each item. The function will then calculate the items remembered within a window of 1 before or 1 after the tested position. The first and last positions must be answered in the correct place.

Usage

```
serial_position(data, position, answer, key, scored, group.by = NULL)
```

Arguments

data	a dataframe of the scored free recall that you would like to calculate - use prop_correct_free() for best formatting.
position	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
answer	a column name of the answer given for that position in the original dataframe.
key	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.
scored	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

This output can then be used to create a serial position curve visualizations, and an example can be found in our manuscript/vignettes.

Important: The code is written assuming group.by variables are between subjects for an individual recall list. If repeated measures are used (i.e., there are multiple lists completed by each participant or multiple list versions), you should use this function several times, once on each list/answer key.

Value

DF_Serial	A dataframe of the proportion correct for each tested position by any optional grouping variables included.

Examples

```
data(free_data)
data(answer_key_free2)

free_data <- subset(free_data,
 List_Type == "Cat_Recall_L1")

DF_long <- arrange_data(data = free_data,
 responses = "Response",
 sep = " ",
 id = "Username")

scored_output <- prop_correct_free(data = DF_long,
 responses = "response",
 key = answer_key_free2$Answer_Key,
 id = "Sub.ID",
 cutoff = 1,
 flag = TRUE,
 group.by = "Version")
```

```
serial_output <- serial_position(data = scored_output$DF_Scored,
 key = answer_key_free2$Answer_Key,
 position = "position",
 scored = "Scored",
 answer = "Answer",
 group.by = "Version")

 head(serial_output)
```

`serial_position_multiple` *Serial Position Calculator for Multiple Lists*

Description

This function calculates the proportion correct of each item in the serial position curve. Data should include the participant's answers in long format (use arrange_data() in this package for help), the answer key of the items in order, and a column that denotes the order a participant listed each item. The function will then calculate the items remembered within a window of 1 before or 1 after the tested position. The first and last positions must be answered in the correct place. Specifically, this function is an extension of serial_position() for free recall when there are multiple lists or randomized lists.

Usage

```
serial_position_multiple(
  data,
  position,
  answer,
  key,
  key.trial,
  id.trial,
  scored,
  group.by = NULL
)
```

Arguments

`data`	a dataframe of the scored free recall that you would like to calculate - use prop_correct_multiple() for best formatting.
`position`	a column name in the dataframe that contains answered position of each response in quotes (i.e., "column")
`answer`	a column name of the answer given for that position in the original dataframe.
`key`	a vector containing the scoring key or data column name. This column does not have to be included in the original dataframe. We assume your answer key is in the tested position order. You should not include duplicates in your answer key.

key.trial	a vector containing the trial numbers for each answer. Note: If you input long data (i.e., repeating trial-answer responses), we will take the unique combination of the responses. If a trial number is repeated, you will receive an error. Key and key.trial can also be a separate dataframe, depending on how your output data is formatted.
id.trial	a column name containing the trial numbers for the participant data from the original dataframe. Note that the free response "key" trial and this trial number should match. The trial key will be repeated for each answer a participant gave.
scored	a column in the original dataframe indicating if the participant got the answer correct (1) or incorrect (0).
group.by	an optional argument that can be used to group the output by condition columns. These columns should be in the original dataframe and concatenated c() if there are multiple columns

Details

This output can then be used to create a serial position curve visualizations, and an example can be found in our manuscript/vignettes.

Value

DF_Serial	A dataframe of the proportion correct for each tested position by any optional grouping variables included.

Examples

```
data("multi_data")
data("multi_answers")

DF_long <- arrange_data(data = multi_data,
                        responses = "Response",
                        sep = " ",
                        id = "Sub.ID",
                        repeated = "List.Number")

library(reshape)
multi_answers$position <- 1:nrow(multi_answers)
answer_long <- melt(multi_answers,
                    measured = colnames(multi_answers),
                    id = "position")
colnames(answer_long) <- c("position", "List.ID", "Answer")

answer_long$List.ID <- gsub(pattern = "List",
                            replacement = "",
                            x = answer_long$List.ID)

DF_long$response <- tolower(DF_long$response)
answer_long$Answer <- tolower(answer_long$Answer)
answer_long$Answer <- gsub(" ", "", answer_long$Answer)
```

```
scored_output <- prop_correct_multiple(data = DF_long,
                                       responses = "response",
                                       key = answer_long$Answer,
                                       key.trial = answer_long$List.ID,
                                       id = "Sub.ID",
                                       id.trial = "List.Number",
                                       cutoff = 1,
                                       flag = TRUE)

head(scored_output$DF_Scored)

head(scored_output$DF_Participant)


serial_output <- serial_position_multiple(data = scored_output$DF_Scored,
                                          position = "position",
                                          answer = "Answer",
                                          key = answer_long$Answer,
                                          key.trial = answer_long$List.ID,
                                          scored = "Scored",
                                          id.trial = "List.Number")

 head(serial_output)
```

`wide_data` *Free Recall Data in Wide Format*

Description

Dataset that includes free recall data in long format. Participants were given a list of words to remember, and then asked to recall the words. This dataset is in wide format, which should be converted with arrange data.

Usage

```
data(wide_data)
```

Format

A data frame of answers for a free recall test data

Sub.ID: the participant id Response: the response the participant gave to the cue Disease.Condition: healthy or sick participant condition

Index